Foundations of Multi-Core Memory Models

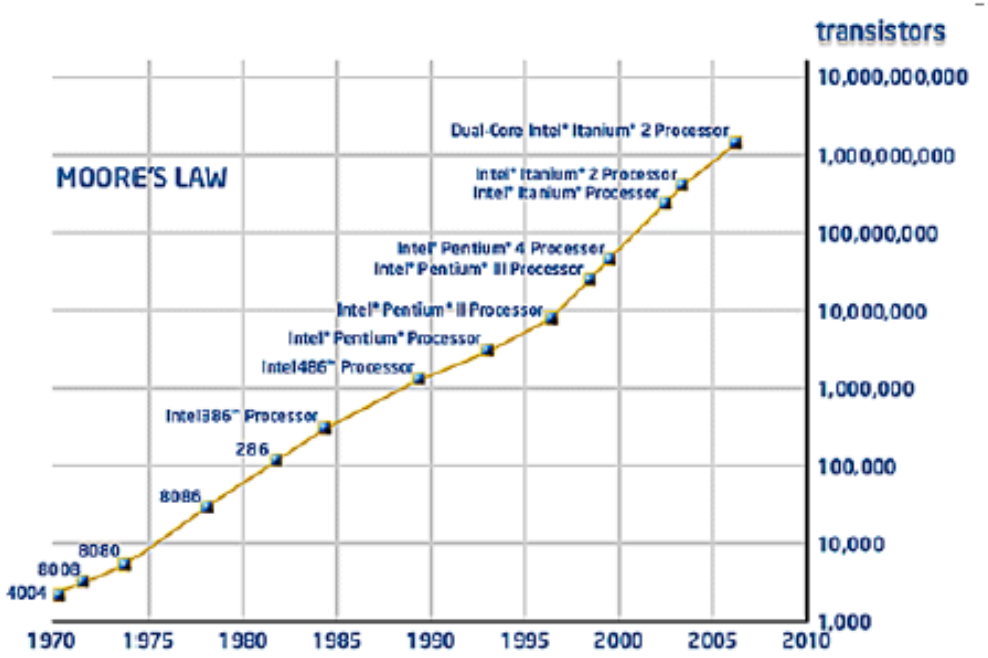
Introductory Meeting to this Seminar

Joost-Pieter Katoen Thomas Noll

Software Modeling and Verification Group

5. April 2011

The engine behind computer power



Moore's Law: Transistor density doubles every 2 years

Concurrent software

In the past ...

- More transistors per chip and faster clock rate
- Same program would execute faster on new processor

Concurrent software

In the past ...

- More transistors per chip and faster clock rate
- Same program would execute faster on new processor

Emerging trend and future

- Parallel hardware (multi-cores)
- Programs must be concurrent
- Applications must be reprogrammed to use parallelism

Concurrent software

In the past ...

- More transistors per chip and faster clock rate
- Same program would execute faster on new processor

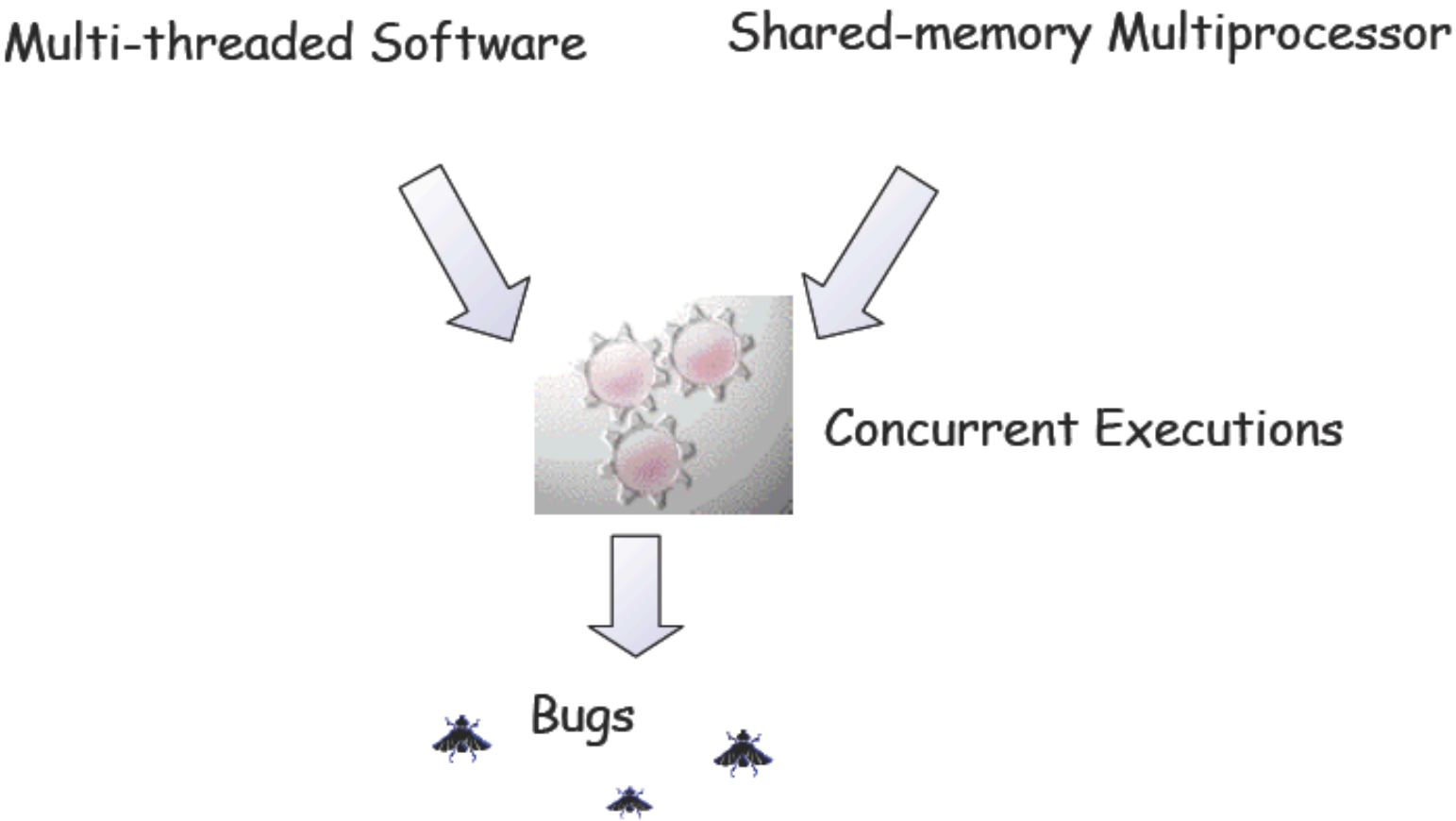
Emerging trend and future

- Parallel hardware (multi-cores)
- Programs must be concurrent
- Applications must be reprogrammed to use parallelism

Herb Sutter, developer of C++, Dr. Dobb's Journal, 2005

The free lunch is over:
A fundamental turn towards concurrency in software

The challenge of concurrent software



Aims of this seminar

Topics

1. Survey of problem domain:
   - state of the practice in multi-core-programming and
   - state of the art in memory consistency models

Aims of this seminar

Topics

1. Survey of problem domain:
   - state of the practice in multi-core-programming and
   - state of the art in memory consistency models
2. Application of concurrency theory:
   - to reveal underlying concepts of parallelism, reordering, causality, and consistency

Aims of this seminar

Topics

1. Survey of problem domain:
   - state of the practice in multi-core-programming and
   - state of the art in memory consistency models
2. Application of concurrency theory:
   - to reveal underlying concepts of parallelism, reordering, causality, and consistency
3. Formal approaches to memory consistency models

Aims of this seminar

Topics

1. Survey of problem domain:
   - state of the practice in multi-core-programming and
   - state of the art in memory consistency models
2. Application of concurrency theory:
   - to reveal underlying concepts of parallelism, reordering, causality, and consistency
3. Formal approaches to memory consistency models
4. Semantics of multithreaded programming

Aims of this seminar

Topics

1. Survey of problem domain:
   - state of the practice in multi-core-programming and
   - state of the art in memory consistency models
2. Application of concurrency theory:
   - to reveal underlying concepts of parallelism, reordering, causality, and consistency
3. Formal approaches to memory consistency models
4. Semantics of multithreaded programming
5. Application of formal verification:
   - to check the correctness of concurrent software

Concurrent software on multiprocessors

Initially x = y = 0

thread 1	thread 2
x = 1	r1 = y
y = 1	r2 = x

Standard Interleavings

x = 1	x = 1	x = 1	r1 = y	r1 = y	r1 = y
y = 1	r1 = y	r1 = y	x = 1	x = 1	r2 = x
r1 = y	y = 1	r2 = x	y = 1	r2 = x	x = 1
r2 = x	r2 = x	y = 1	r2 = x	y = 1	y = 1
r1=r2=1	r1=0,r2=1	r1=0,r2=1	r1=0,r2=1	r1=0,r2=1	r1=r2=0

Concurrent software on multiprocessors

Initially x = y = 0

thread 1	thread 2
x = 1	r1 = y
y = 1	r2 = x

Standard Interleavings

x = 1	x = 1	x = 1	r1 = y	r1 = y	r1 = y
y = 1	r1 = y	r1 = y	x = 1	x = 1	r2 = x
r1 = y	y = 1	r2 = x	y = 1	r2 = x	x = 1
r2 = x	r2 = x	y = 1	r2 = x	y = 1	y = 1
r1=r2=1	r1=0,r2=1	r1=0,r2=1	r1=0,r2=1	r1=0,r2=1	r1=r2=0

Can we conclude that if $r1{=}1$ then $r2$ must be 1?

Concurrent software on multiprocessors

Initially x = y = 0

thread 1	thread 2
x = 1	r1 = y
y = 1	r2 = x

Standard Interleavings

x = 1	x = 1	x = 1	r1 = y	r1 = y	r1 = y
y = 1	r1 = y	r1 = y	x = 1	x = 1	r2 = x
r1 = y	y = 1	r2 = x	y = 1	r2 = x	x = 1
r2 = x	r2 = x	y = 1	r2 = x	y = 1	y = 1
r1=r2=1	r1=0,r2=1	r1=0,r2=1	r1=0,r2=1	r1=0,r2=1	r1=r2=0

Can we conclude that if $r1{=}1$ then $r2$ must be 1?

No! On modern multiprocessors, $r1{=}1$ and $r2{=}0$ is possible.

Concurrent software on multiprocessors

Fact

A modern multiprocessor (or: compiler) does not enforce global ordering of all instructions for performance reasons.

Concurrent software on multiprocessors

Fact

A modern multiprocessor (or: compiler) does not enforce global ordering of all instructions for performance reasons.

Sequential consistency (SC) [Lamport 79]

A multiprocessor system is sequentially consistent if the result of an execution is the same as if:

1. the operations of all processors were executed in some sequential order, and
2. the operations of each individual processor appear in this sequence in the order specified by its program.

So, under SC, the previous example establishes that $r1{=}1$ implies $r2{=}1$.

Concurrent software on multiprocessors

Fact

A modern multiprocessor (or: compiler) does not enforce global ordering of all instructions for performance reasons.

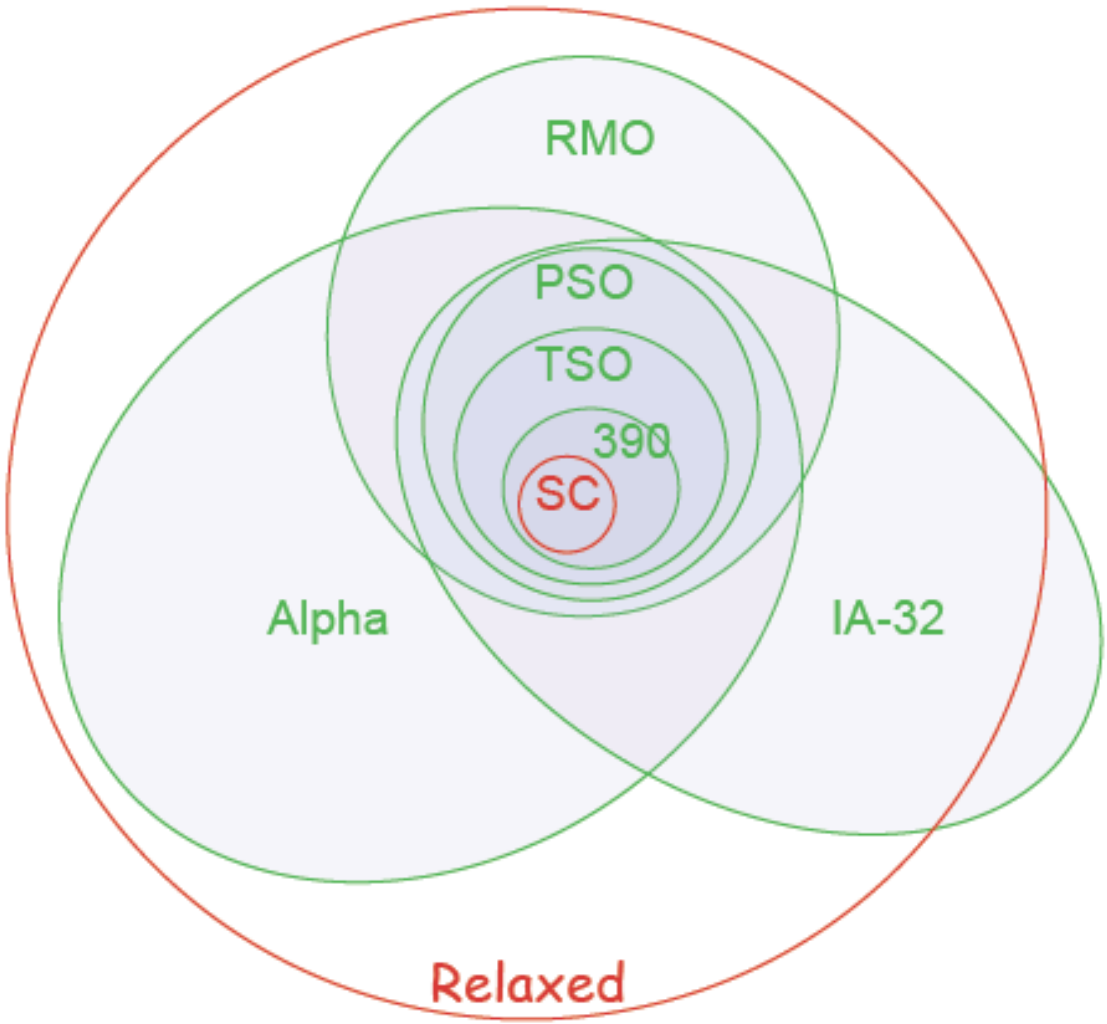
Practice: SC is too restrictive

Relax the sequential consistency assumption:

1. In theory:
   - TSO (total store ordering)
   - PSO (partial store ordering)
   - RMO (relaxed memory ordering), ...
2. In practice: Alpha, Intel x86, IBM 370, Sun SPARC, ARM, ...

Under these "models", $r1{=}1$ implies $r2{=}1$ is not guaranteed.

Memory models: overview



Memory models: animation

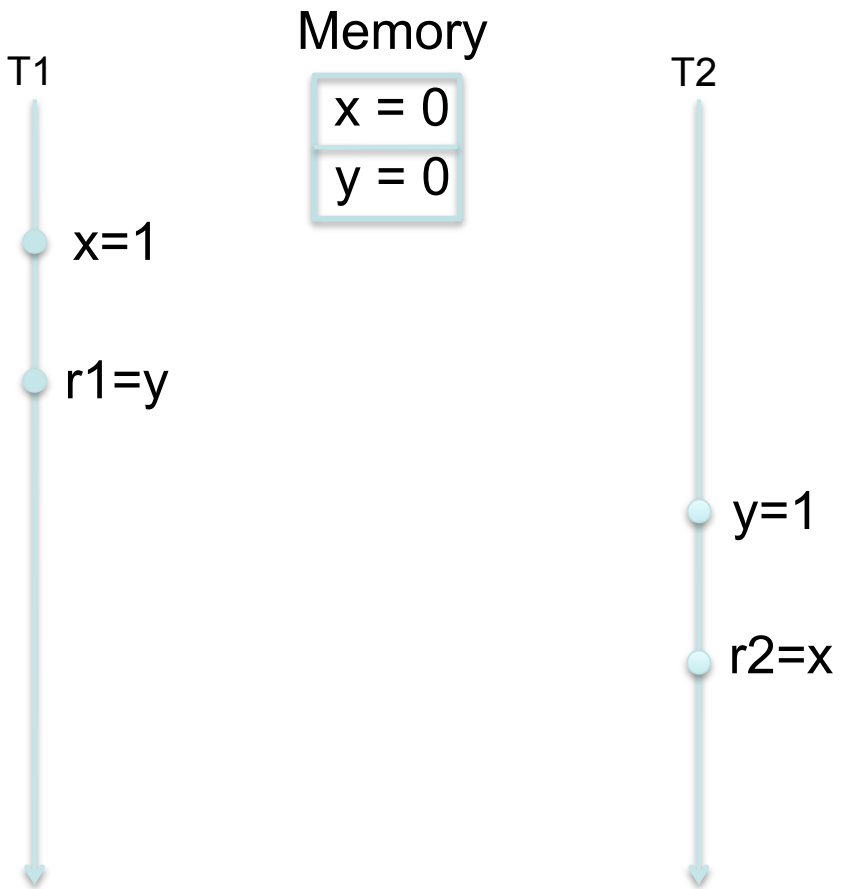
An illustrative example

Initially: x = y = 0

thread1:	thread2:
1: x = 1	3: y = 1
2: r1 = y	4: r2 = x

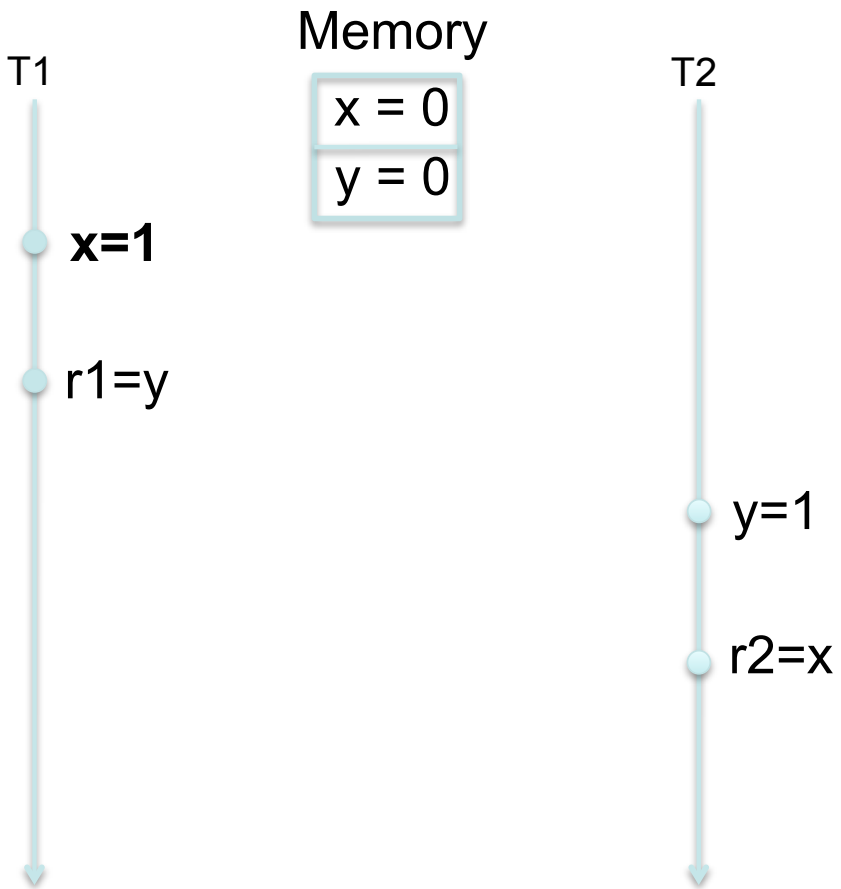
Memory models: animation

Sequential Consistency (SC)



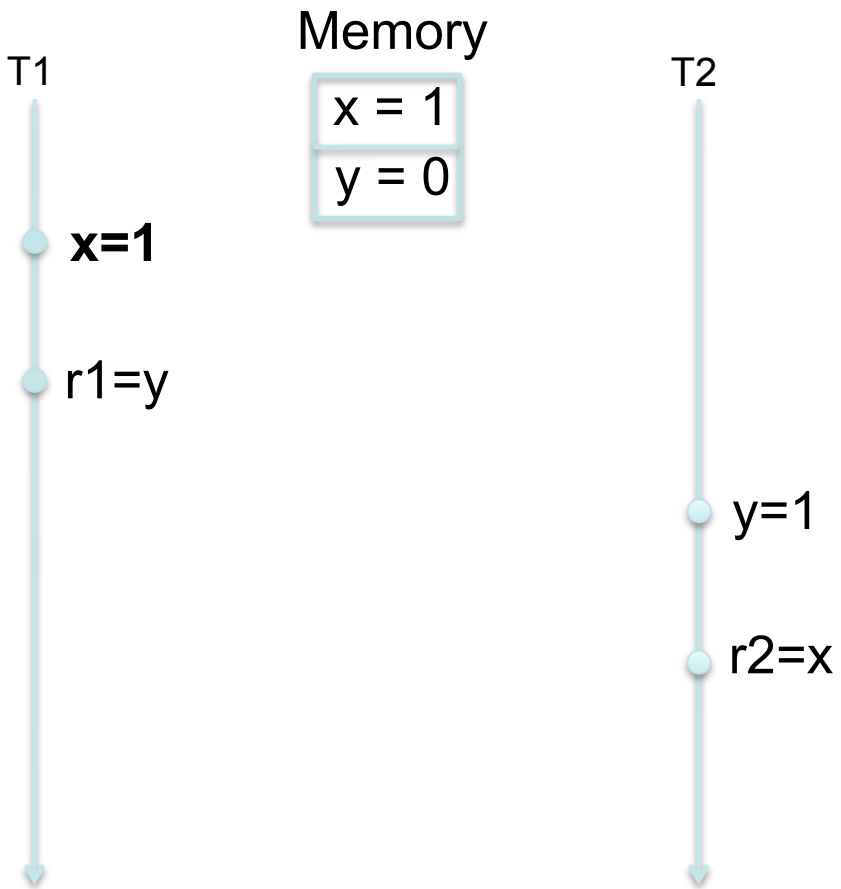
Memory models: animation

Sequential Consistency (SC)



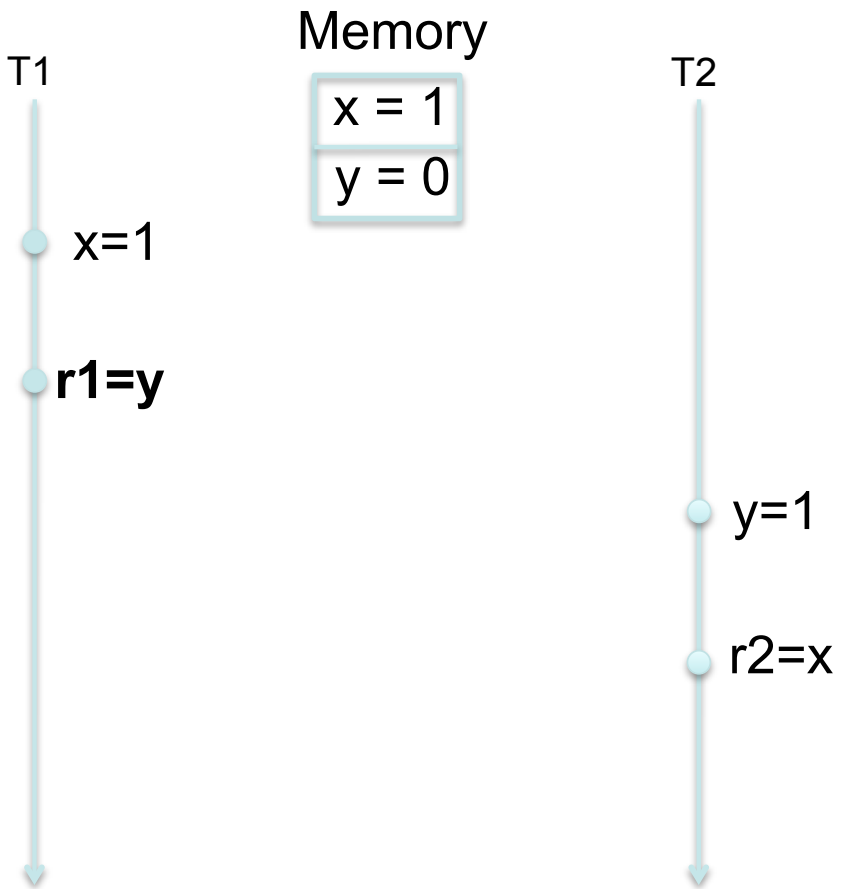
Memory models: animation

Sequential Consistency (SC)



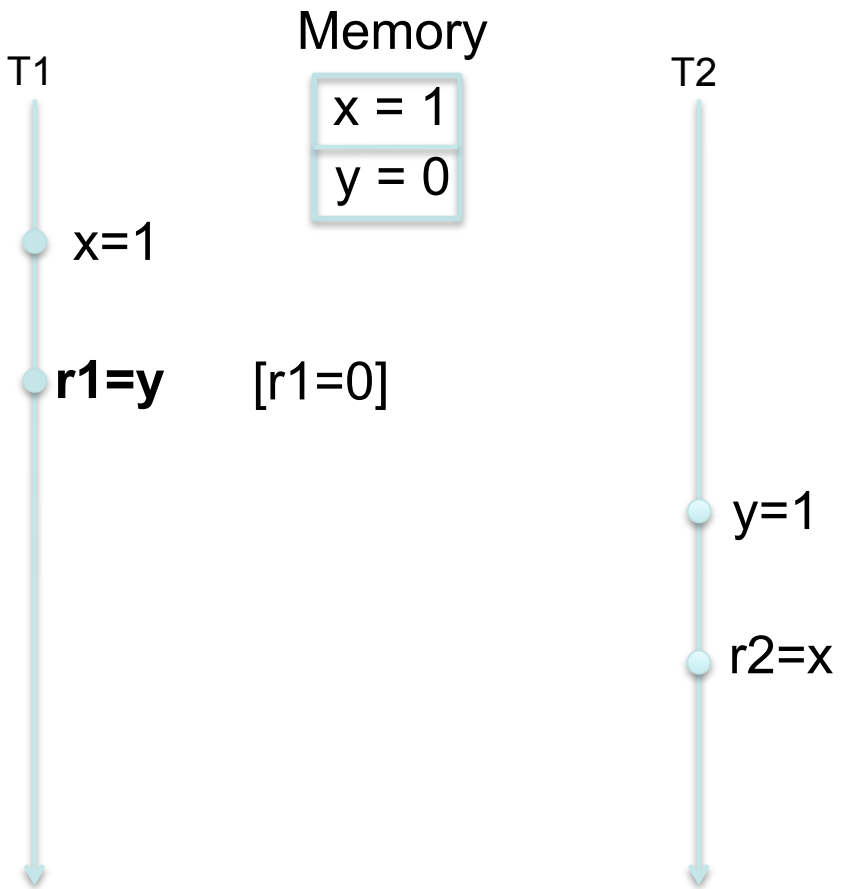
Memory models: animation

Sequential Consistency (SC)



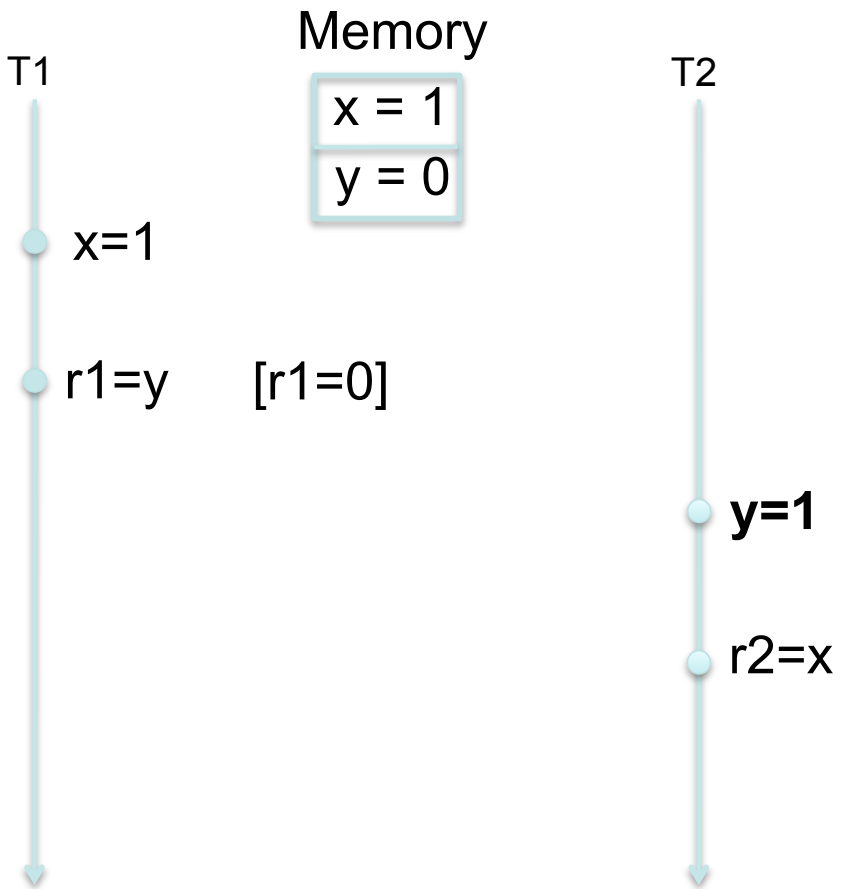
Memory models: animation

Sequential Consistency (SC)



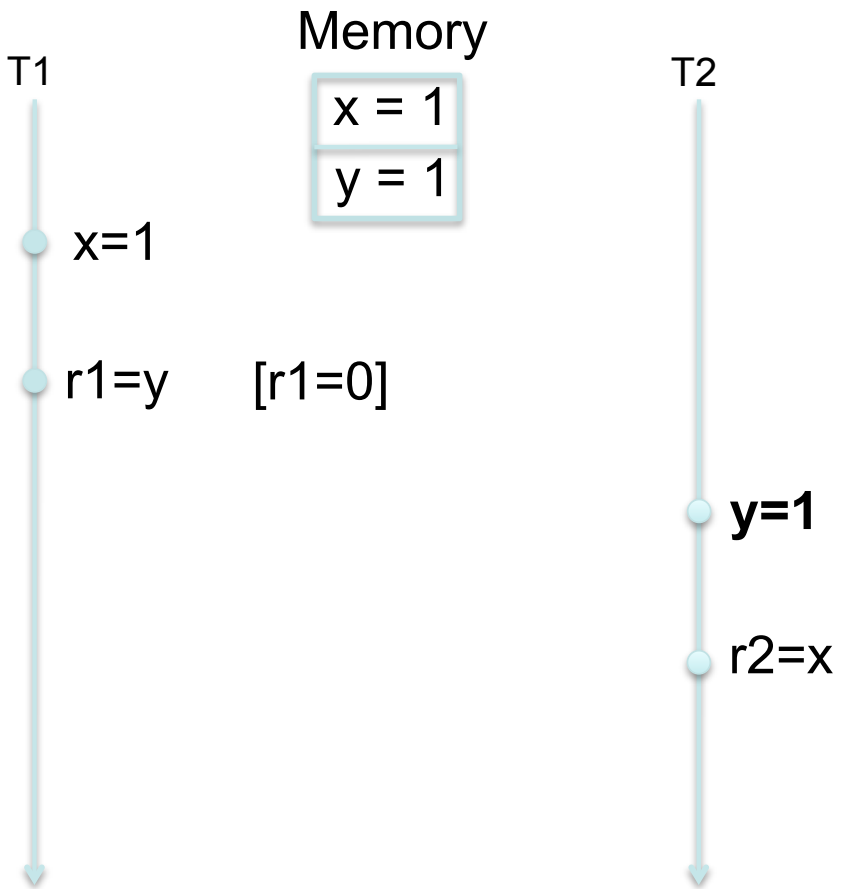
Memory models: animation

Sequential Consistency (SC)



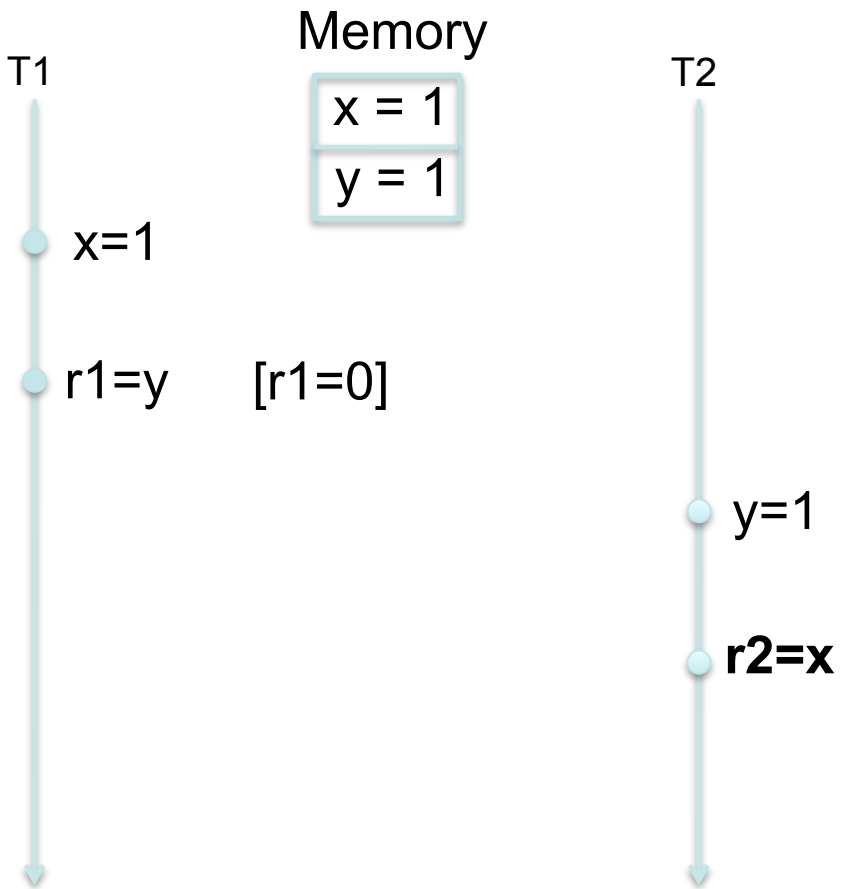
Memory models: animation

Sequential Consistency (SC)



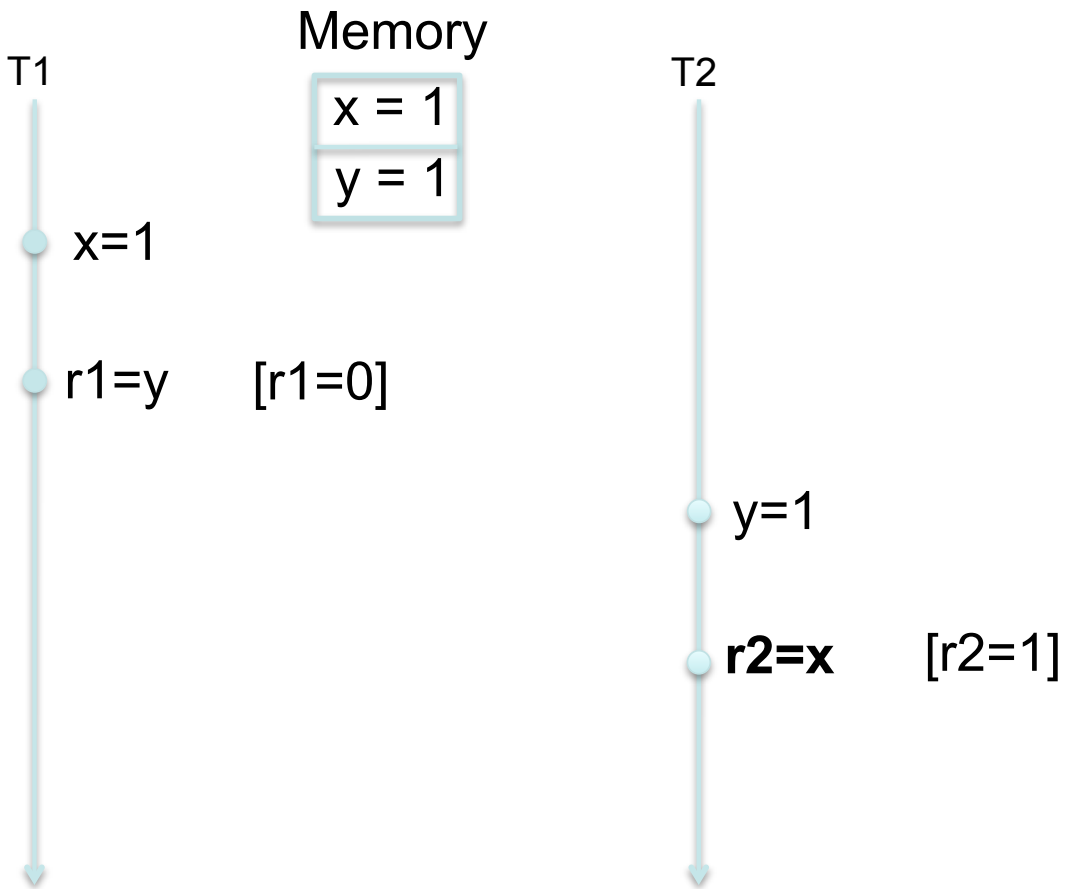
Memory models: animation

Sequential Consistency (SC)



Memory models: animation

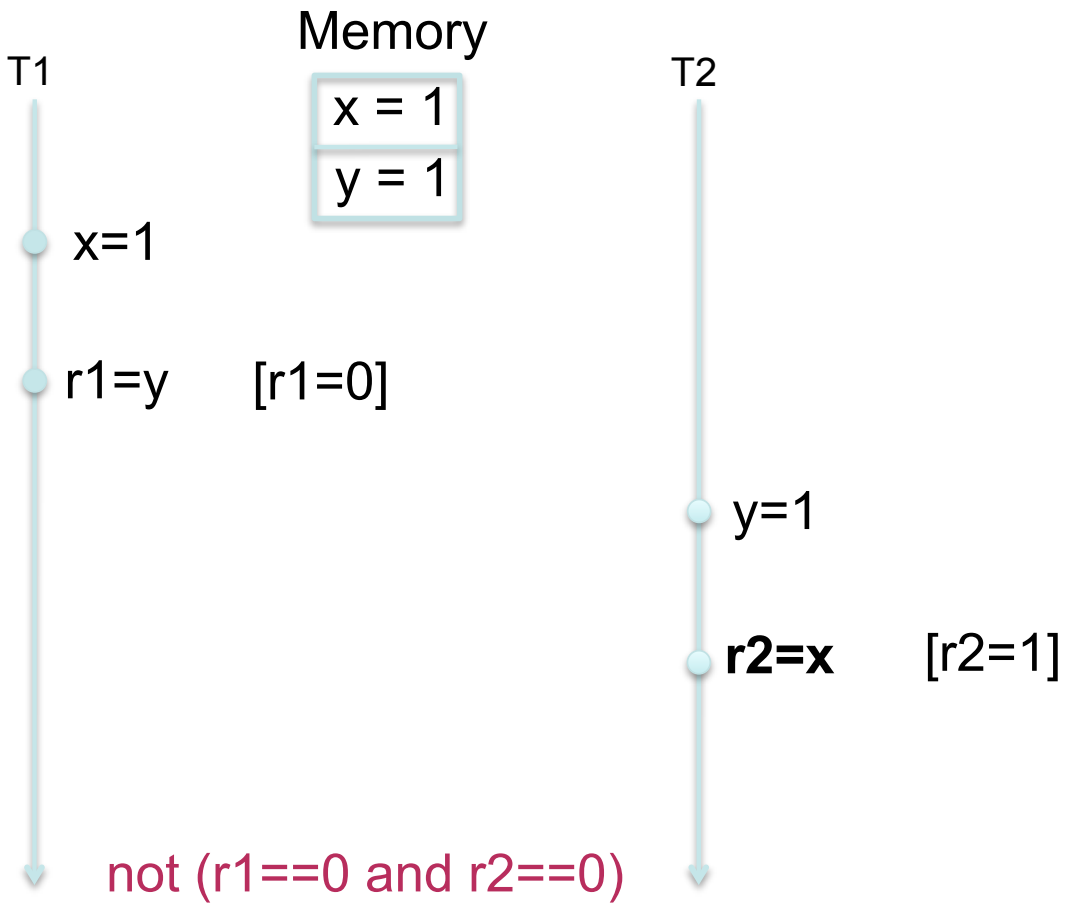
Sequential Consistency (SC)



2

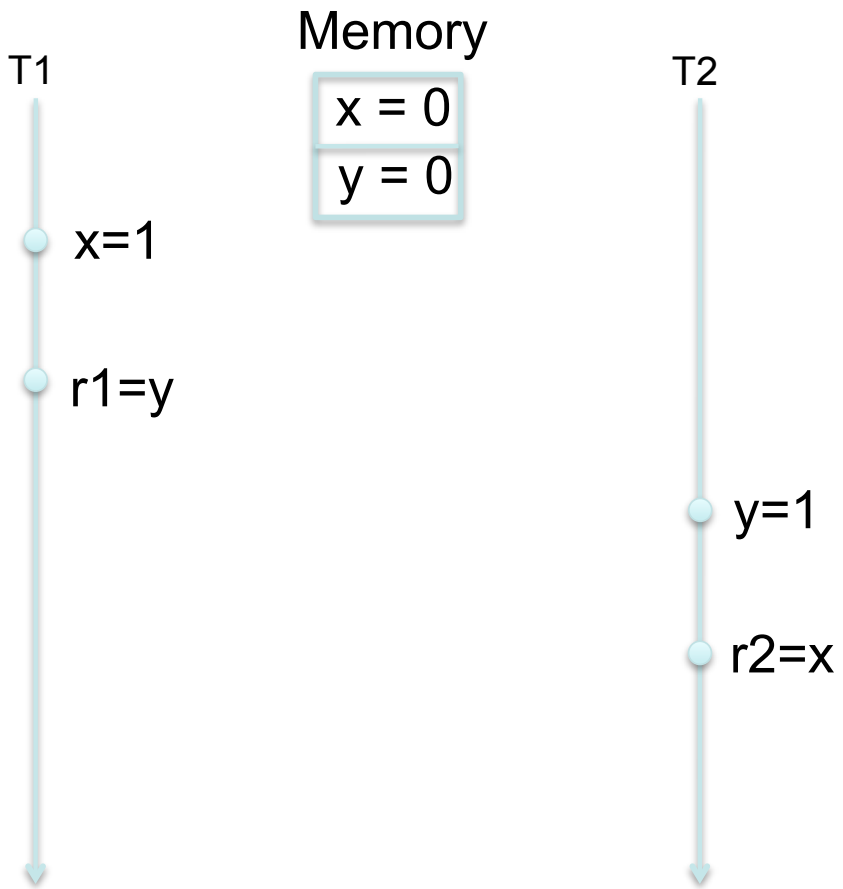
Memory models: animation

Sequential Consistency (SC)



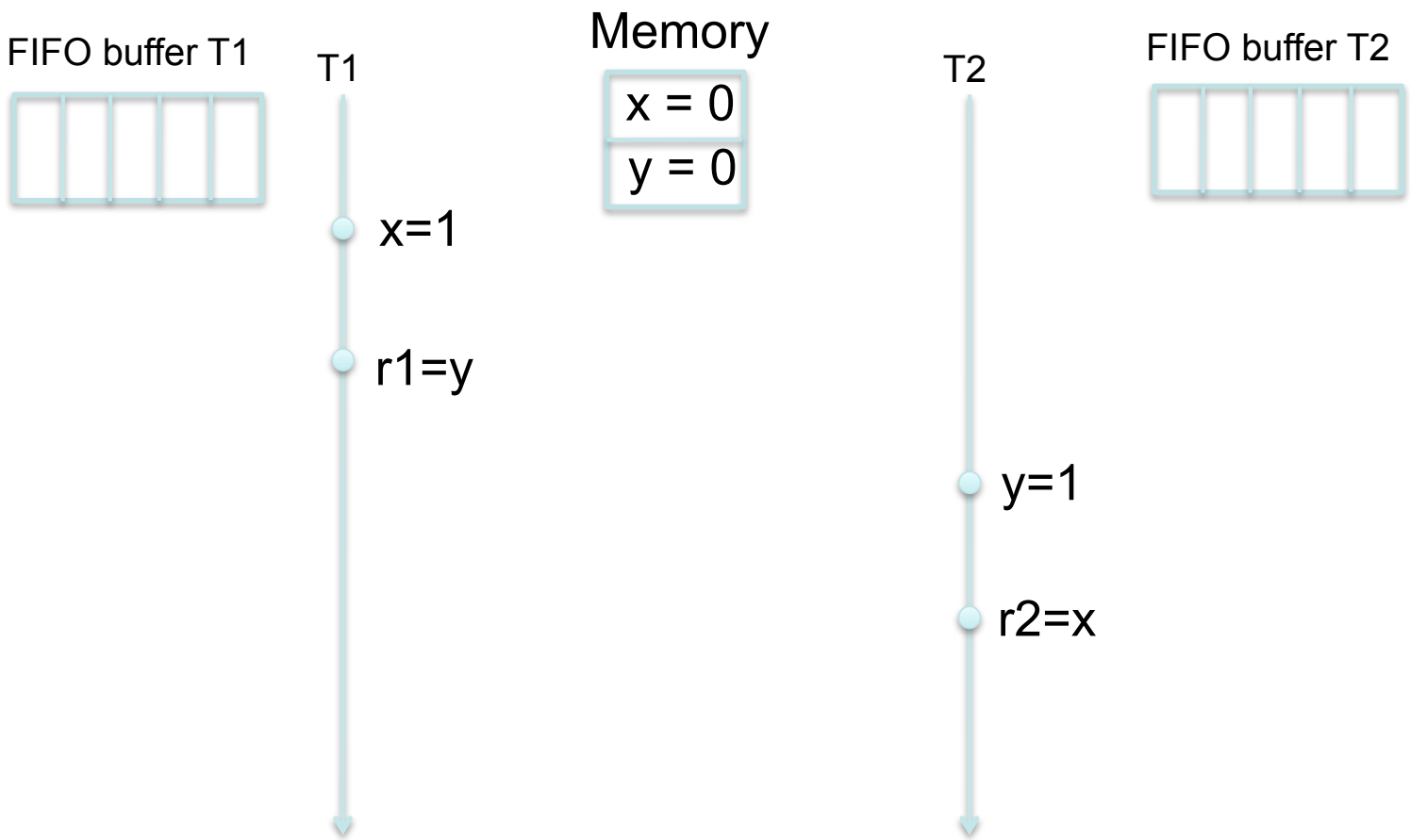
Memory models: animation

Total Store Ordering (TSO)



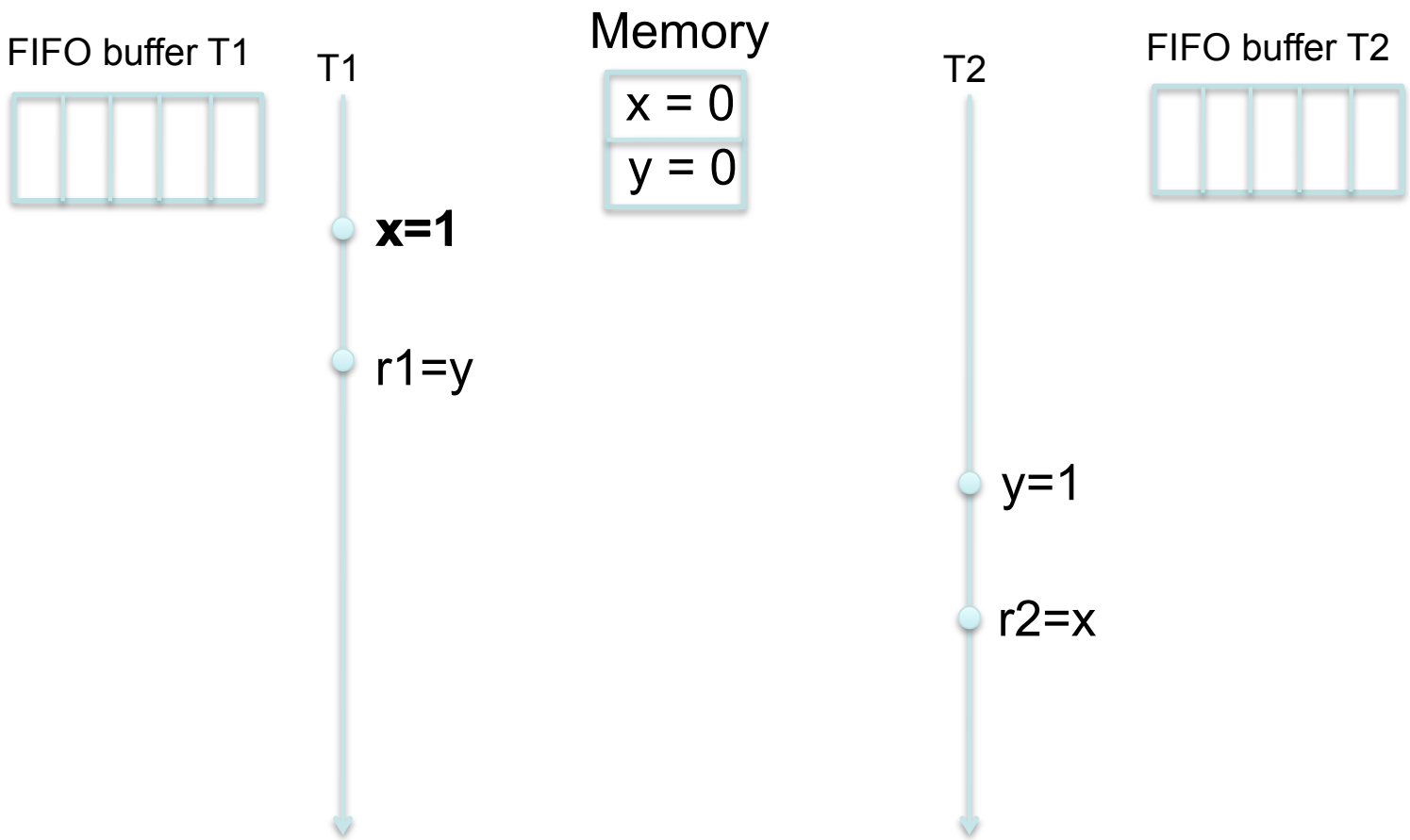
Memory models: animation

Total Store Ordering (TSO)



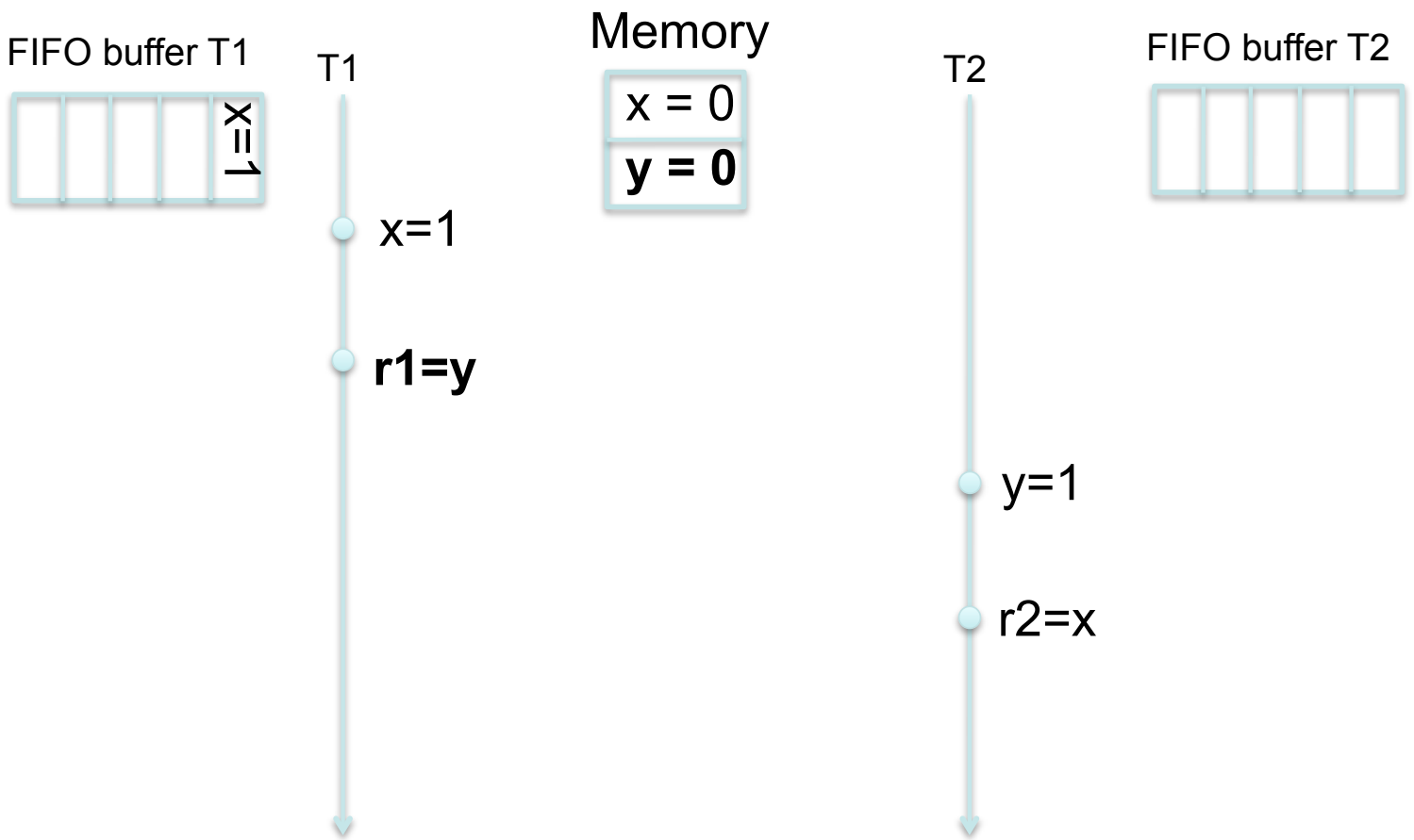
Memory models: animation

Total Store Ordering (TSO)



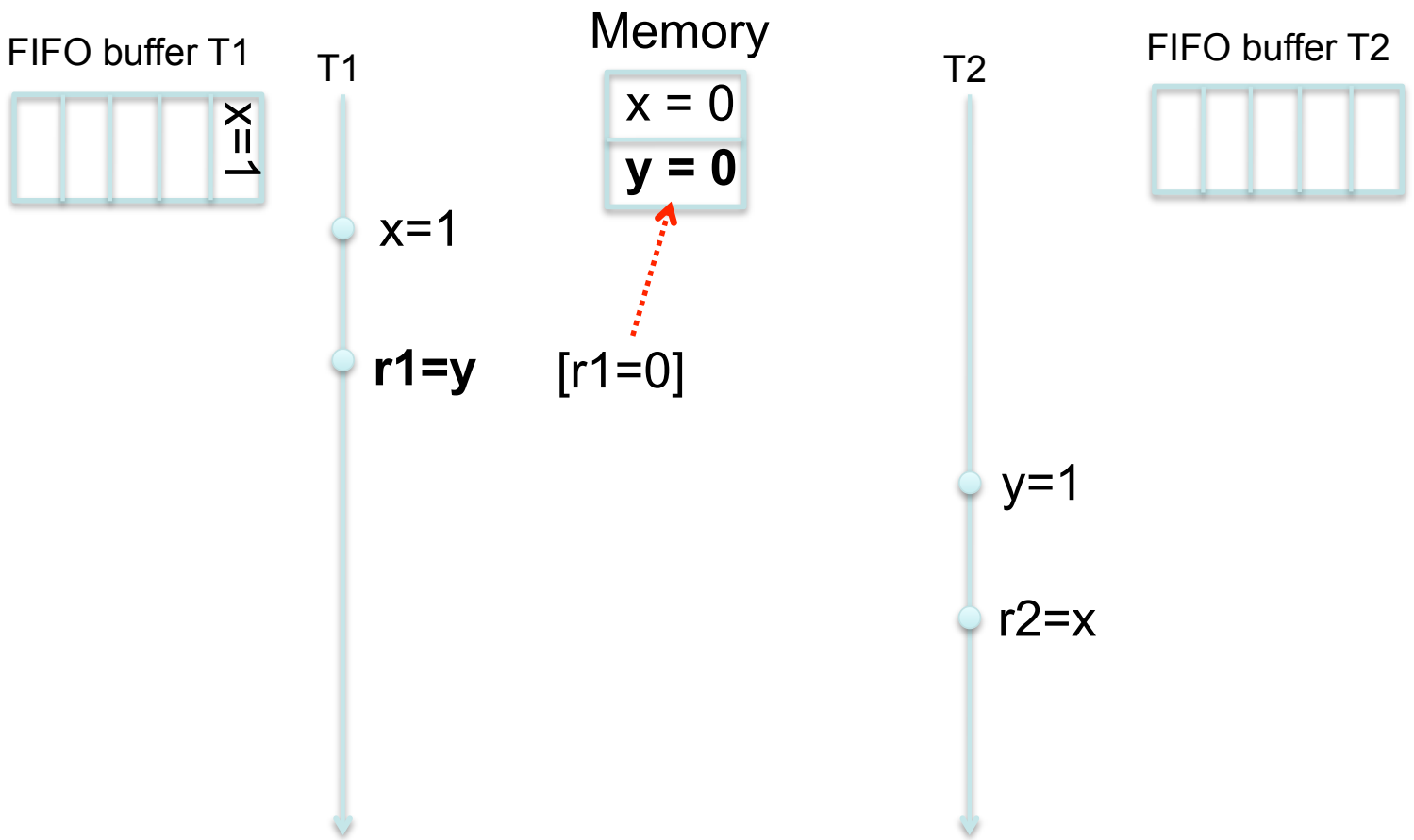
Memory models: animation

Total Store Ordering (TSO)



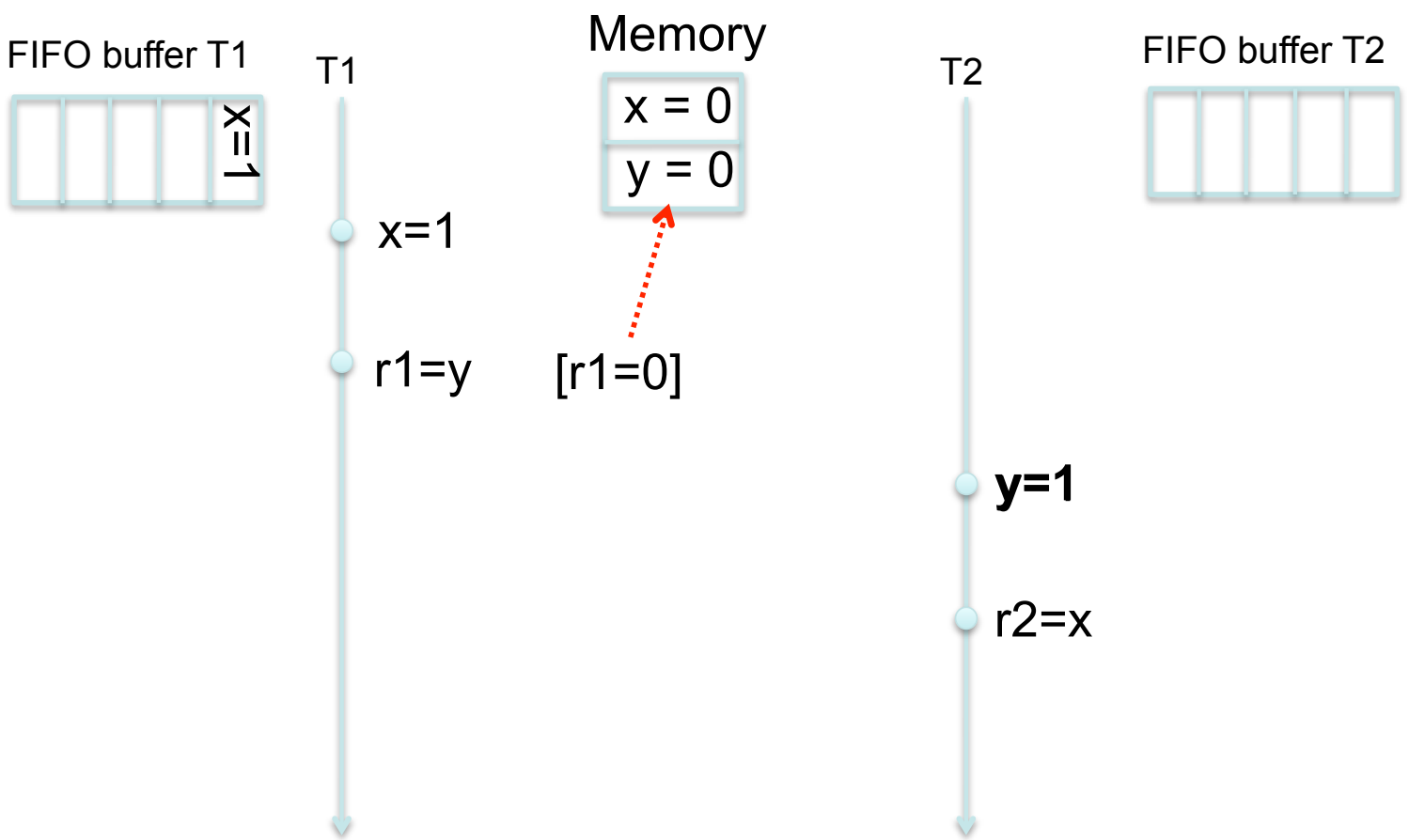
Memory models: animation

Total Store Ordering (TSO)



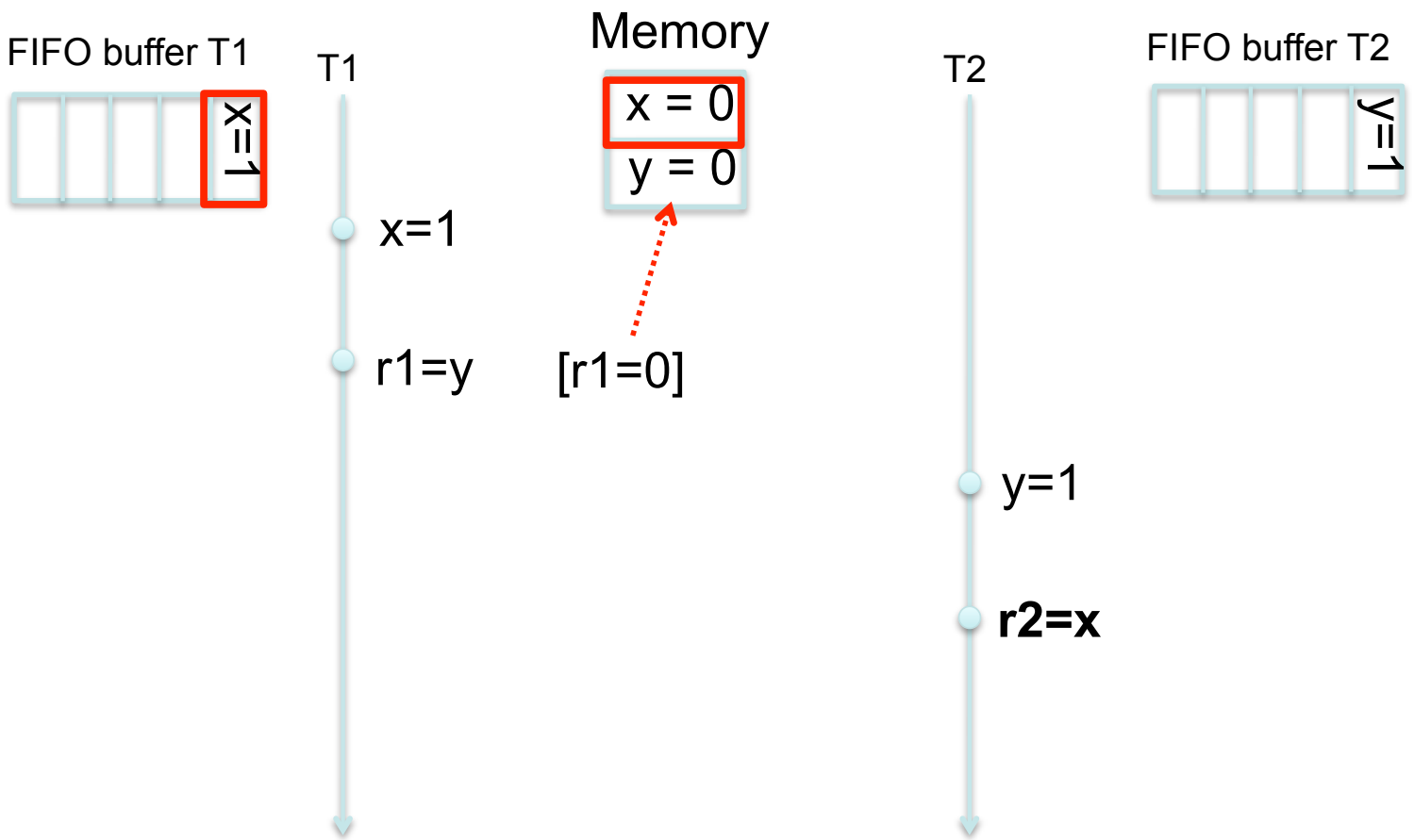
Memory models: animation

Total Store Ordering (TSO)

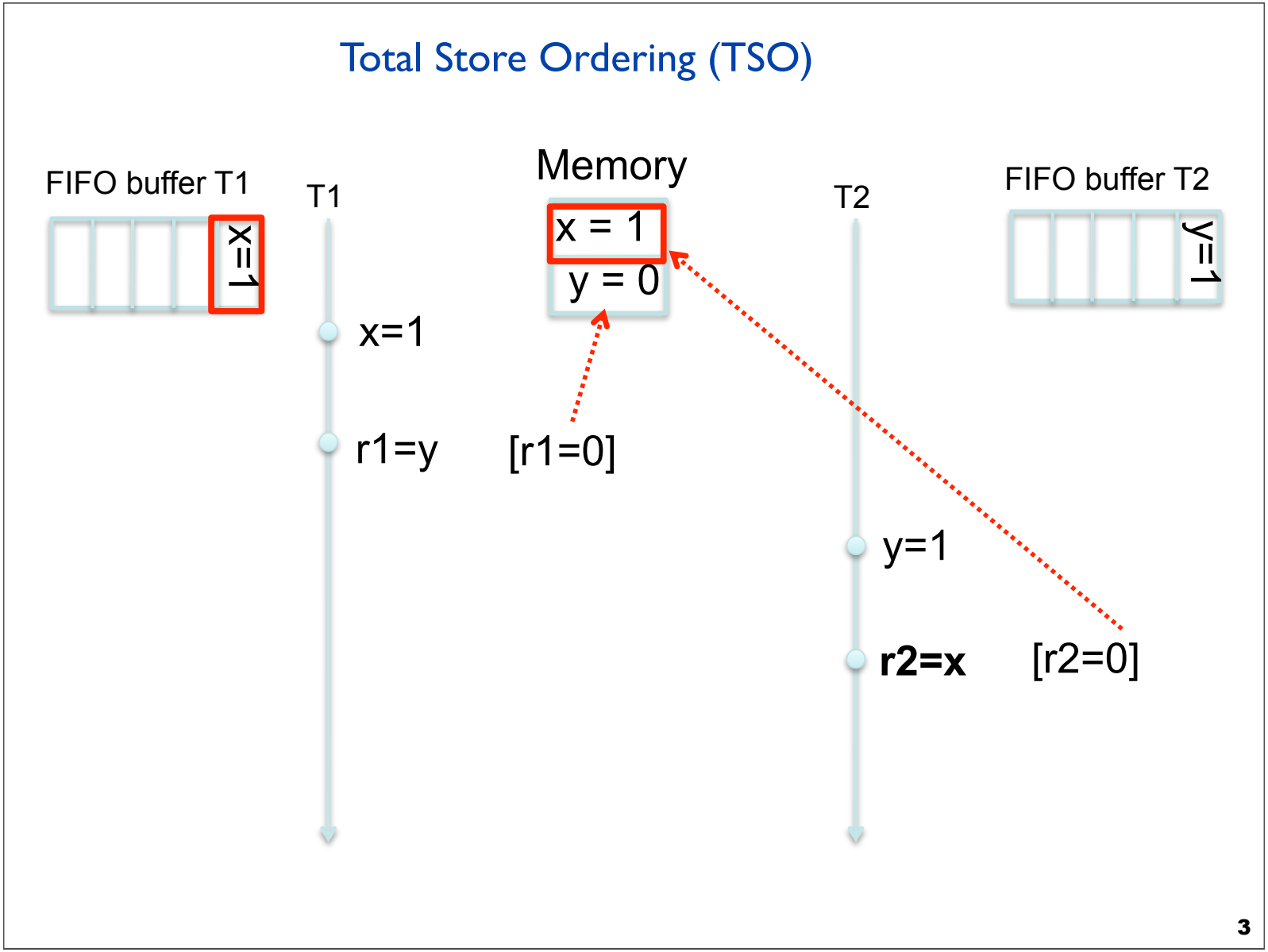


Memory models: animation

Total Store Ordering (TSO)

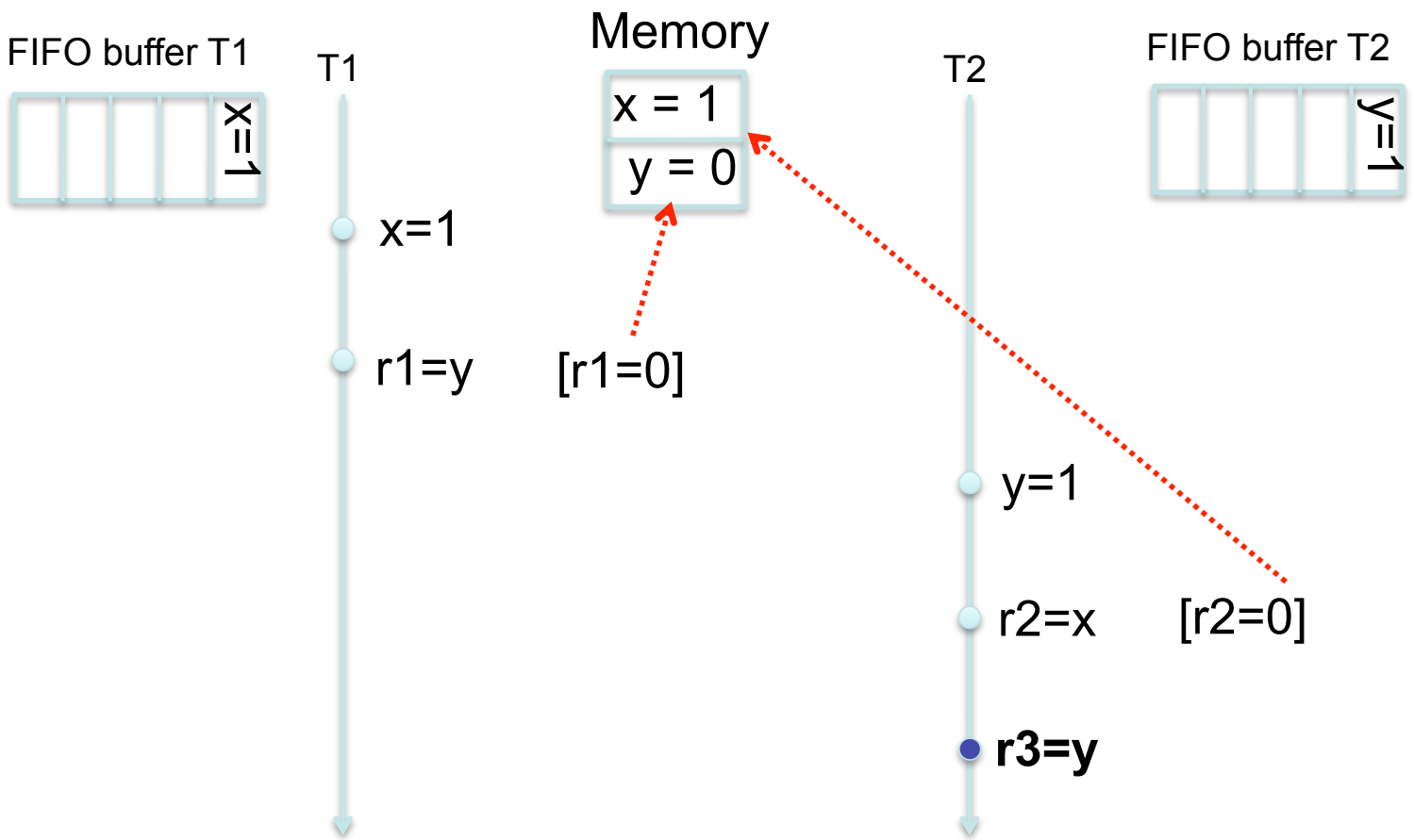


Memory models: animation

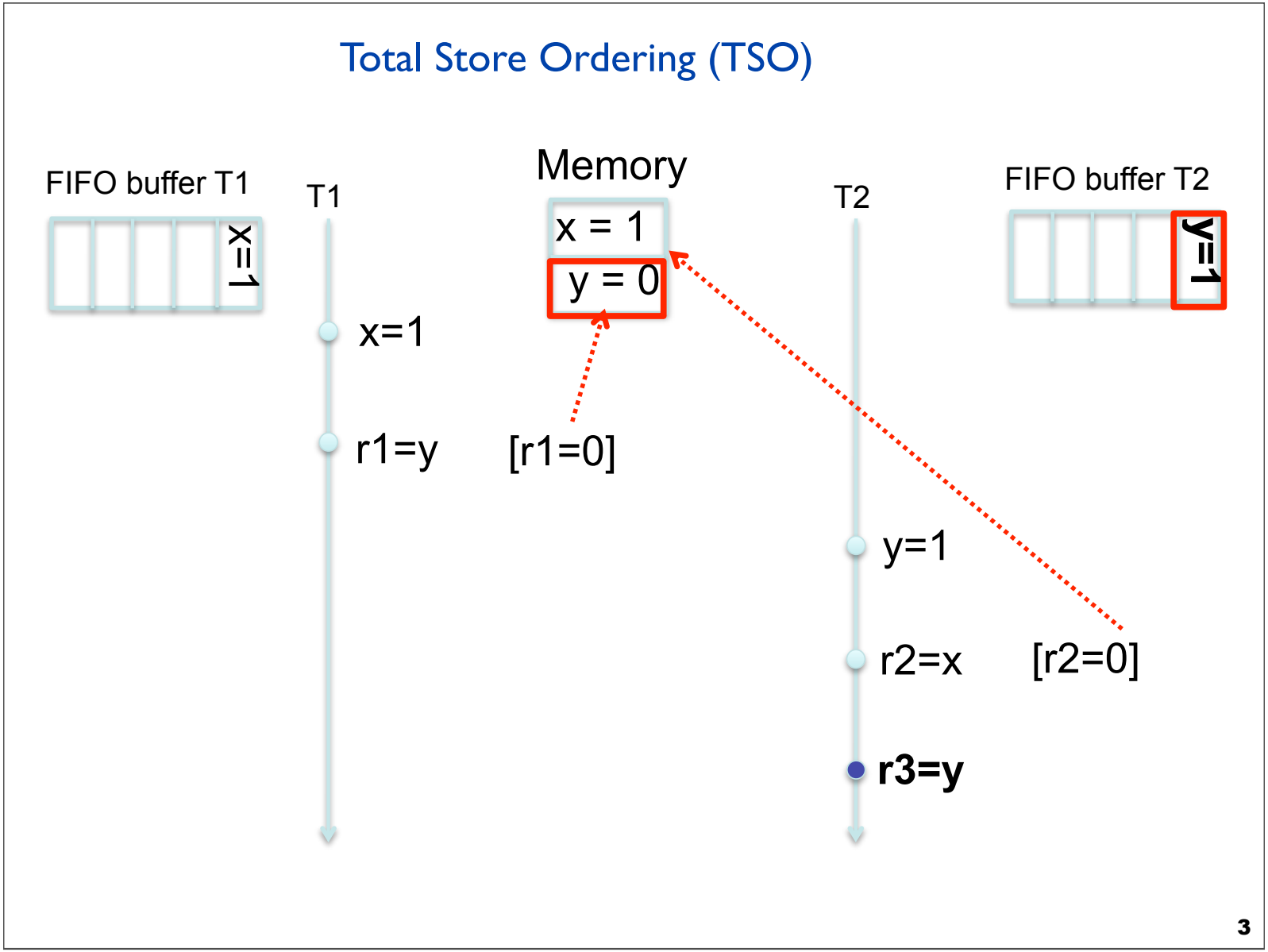


Memory models: animation

Total Store Ordering (TSO)

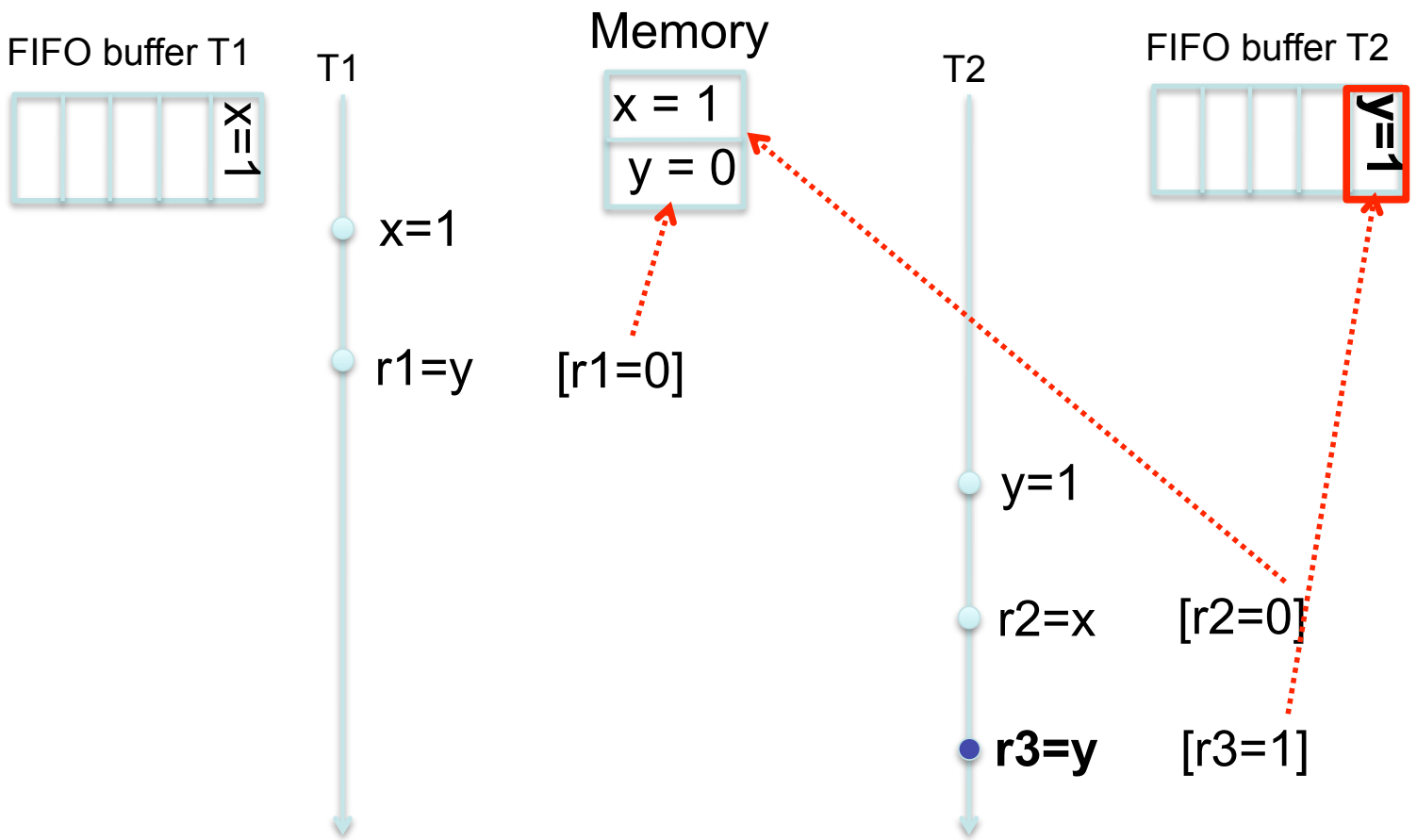


Memory models: animation



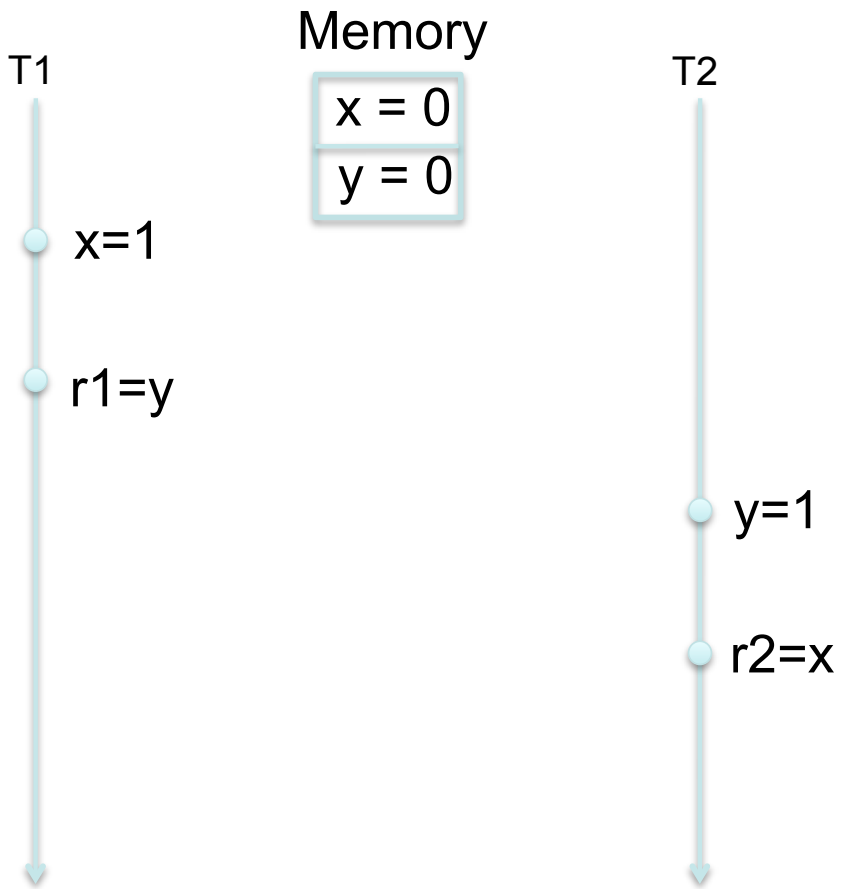
Memory models: animation

Total Store Ordering (TSO)



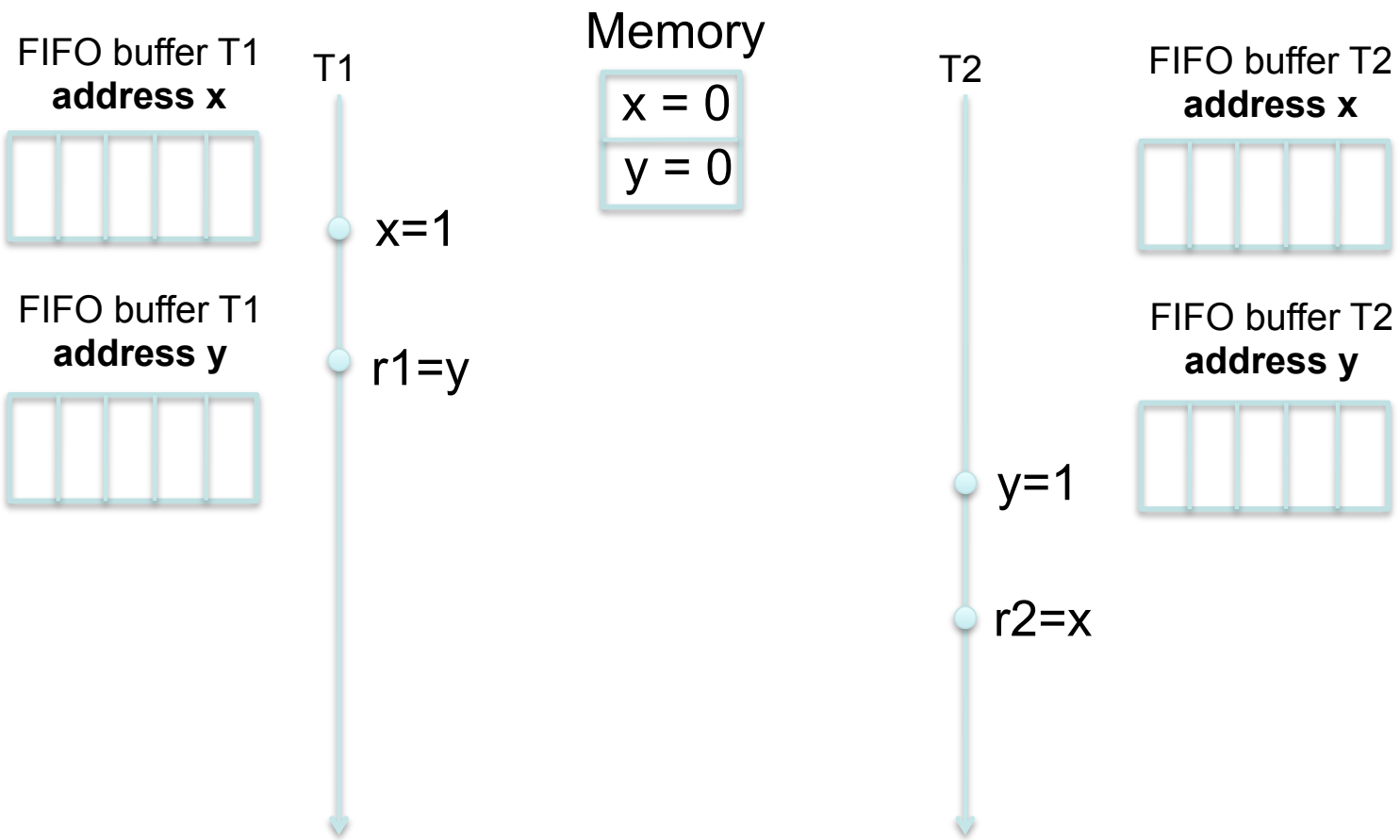
Memory models: animation

Partial Store Ordering (PSO)



Memory models: animation

Partial Store Ordering (PSO)



Goals

Aims of this seminar

- Independent understanding of a scientific topic
- Acquiring, reading and understanding scientific literature
- Writing of your own report on a scientific topic
- Oral presentation of your results

Requirements on your report

Your report

- independent writing of a report of about 20 pages
- complete set of references to all consulted literature
- correct citation of important literature
- Plagiarism:
  Taking text blocks (from literature or web) without source indication causes immediate exclusion from this seminar
- Font size 12pt with "normal" page layout
- Language: German or English
- We expect the correct usage of spelling and grammar:
  $\geq$ 10 errors per page $\Longrightarrow$ abortion of correction

Requirements on your talk

Your talk

- Talk of about 45 minutes
- Focus your talk on the audience
- Your slides:
  - $\leq$ 15 lines of text
  - use (base) colors in a useful manner
- Language: German or English
- No spelling mistakes please!
- Finish in time. Overtime is bad.
- Ask for questions.
- Elaborate on previous questions.

Final prepations

Preparation of your talk

- Setup laptop and projector ahead of time.
- Use a pointer.
- Use links in your slides.
- Number your slides.
- Multiple copies: laptop, USB, web.
- Have backup slides ready for expected questions.

Important dates

Talks

The seminar will be held as a blockseminar on October 4 and 5, 2011.

Important dates

Talks

The seminar will be held as a blockseminar on October 4 and 5, 2011.

Deadlines

You are requested to adhere to the following firm deadlines:

- now: obtain the required literature from the web or library
- ten weeks before your talk: present a table of contents
- eigth weeks before your talk: preliminary version of your report
- four weeks before your talk: final version of your report
- two weeks before your talk: preliminary version of your slides
- one week before your talk: final version of your slides

Important dates

Talks

The seminar will be held as a blockseminar on October 4 and 5, 2011.

Deadlines

You are requested to adhere to the following firm deadlines:

- now: obtain the required literature from the web or library
- ten weeks before your talk: present a table of contents
- eigth weeks before your talk: preliminary version of your report
- four weeks before your talk: final version of your report
- two weeks before your talk: preliminary version of your slides
- one week before your talk: final version of your slides

Missing a deadline causes immediate exclusion from this seminar.

Selecting your topic

- You obtain(ed) a list of topics of this seminar.
- Indicate the preference of your topics (first, second, third).
- We do our best to find an adequate topic-student distribution.
- Disclaimer: no guarantee for an optimal solution.
- Your topic will be published on our website by April 11.
- Then also your supervisor will be indicated.

You have upto three weeks to refrain from participating in this seminar.

Later cancellation (by you or by us) causes a not passed for this seminar and reduces your (three) possibilities by one.

Introduction: topic 1

Topic 1

The following papers give an introduction to the issues arising with shared-memory computations, ordered chronologically. This material is intended for a single seminar.

(1a) Leslie Lamport: How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Trans. Computers 28(9): 690-691 (1979)

(1b) Sarita V. Adve, Kourosh Gharachorloo: Shared memory consistency models: A tutorial. IEEE Computer 29(12): 66-76 (1996)

(1c) Sarita V. Adve, Hans-Juergen Boehm: Memory models: a case for rethinking parallel languages and hardware. Commun. ACM 53(8): 90-101 (2010)

Historical account: topics 2–4

Topic 2

Lamport presents two theorems for a shared memory setting in which one process at a time can modify the data, but concurrent readings and writings are possible. Misra presents the weakest axioms for a similar setting to facilitate correctness proofs of shared memory programs.

(2a) Leslie Lamport: Concurrent Reading and Writing. Commun. ACM 20(11): 806-811 (1977)

(2b) Jayadev Misra: Axioms for Memory Access in Asynchronous Hardware Systems. ACM Trans. Program. Lang. Syst. 8(1): 142-153 (1986)

Historical account: topics 2–4

Topic 3

Shasha and Snir describe an approach to determine the minimal delay within a process such as to accommodate sequential consistency.

(3) Dennis Shasha, Marc Snir: Efficient and Correct Execution of Parallel Programs that Share Memory. ACM Trans. Program. Lang. Syst. 10(2): 282-312 (1988)

Historical account: topics 2–4

Topic 4

Ahamad et al. give a first formalisation of a so-called weak memory model. Such model allows for more concurrent memory accesses than for sequentially consistent memories. Besides the formalisation, an implementation is described in detail.

(4) Mustaque Ahamad, Gil Neiger, James E. Burns, Prince Kohli, Phillip W. Hutto: Causal Memory: Definitions, Implementation, and Programming. Distributed Computing 9(1): 37-49 (1995)

Semantics: topics 5–10

Topic 5

Manson et al. define the semantics of multi-threaded Java programs and partially determines legal implementations of Java virtual machines and compilers. It provides a simple interface for correctly synchronized programs – it guarantees sequential consistency to data-race-free programs. Flaws in this model were found and (partially) repaired by Aspinall and Sevcik.

(5a) Jeremy Manson, William Pugh, Sarita V. Adve: The Java memory model. POPL 2005: 378-391

(5b) David Aspinall, Jaroslav Sevcik: Java Memory Model Examples: Good, Bad and Ugly. VAMP 2007

Semantics: topics 5–10

Topic 6

Saraswat et al. establish that all models in their mathematical framework satisfy the following (fundamental) property of relaxed memory models: programs whose sequentially consistent (SC) executions have no races must have have only SC executions.

(6) Vijay A. Saraswat, Radha Jagadeesan, Maged M. Michael, Christoph von Praun: A theory of memory models. PPOPP 2007: 161-172.

Semantics: topics 5–10

Topics 7+8

Recently, Jagadeesan and co-workers presented an attempt to fit relaxed memory models into structured operational semantics and showed that the basic properties of the JMM hold in their formalisation.

(7) Radha Jagadeesan, Corin Pitcher, James Riely: Generative Operational Semantics for Relaxed Memory Models. ESOP 2010: 307-326.

An alternative approach is followed by Boudol and Petri:

(8) Gerard Boudol, Gustavo Petri: Relaxed memory models: an operational approach. POPL 2009: 392-403.

Semantics: topics 5–10

Topic 9

A popular technique in “optimisingcompilers for multi-core machines is so-called speculative re-ordering of instructions. This allows for a speed up of the program execution as it allows more concurrency. Boudol and Petri recently gave a first formalisation of this technique.

(9) Gerard Boudol, Gustavo Petri: A Theory of Speculative Computation. ESOP 2010: 165-184

Semantics: topics 5–10

Topic 10

The last two papers of this section (intended for a single seminar) present two approaches towards the mathematical semantics of C++ concurrency.

(10a) Mark Batty, Scott Owens, Susmit Sarkar, Peter Sewell, Tjark Weber: Mathematizing C++ concurrency. POPL 2011: 55-66

(10b) Hans-Juergen Boehm, Sarita V. Adve: Foundations of the C++ concurrency memory model. PLDI 2008: 68-78

Semantics: topics 11–14

Topic 11

Brinksma gives a process algebraic account to proving the correctness of a lazy caching algorithm. The proof is based on refinement.

(11a) Rob Gerth: Sequential Consistency and the Lazy Caching Algorithm. Distributed Computing 12(2-3): 57-59 (1999)

(11b) Ed Brinksma: Cache Consistency by Design. Distributed Computing 12(2-3): 61-74 (1999)

Semantics: topics 11–14

Topic 12

Model checking transactional memories (TMs) is difficult because of the unbounded number, length, and delay of concurrent transactions, as well as the unbounded size of the memory. Recently, Guerraoui et al. showed that, under certain conditions satisfied by most TMs, the model checking problem can be reduced to a finite-state problem.

(12) Rachid Guerraoui, Thomas A. Henzinger, Vasu Singh: Model checking transactional memories. Distributed Computing 22(3): 129-145 (2010)

Semantics: topics 11–14

Topic 13

Atig et al. define abstract operational models for three different weak memory models. Their models are based on state machines with (potentially unbounded) FIFO buffers. The authors investigate the decidability of their reachability and their repeated reachability problems.

(13) Mohamed Faouzi Atig, Ahmed Bouajjani, Sebastian Burckhardt, Madanlal Musuvathi: On the verification problem for weak memory models. POPL 2010: 7-18

Semantics: topics 11–14

Topic 14

Microsoft developed a monitor algorithm that can detect the presence of program executions that are not sequentially consistent due to store buffers. Such monitor is combined with a stateless model checker that verifies that every sequentially consistent execution is correct. Practical results are provided.

(14) Sebastian Burckhardt, Madanlal Musuvathi: Effective Program Verification for Relaxed Memory Models. CAV 2008: 107-120

Some hints

Hints

- Take your time to understand your literature.
- Be proactive! Look for additional literature and information.
- Discuss the content of your report with other students.
- Be proactive! Contact your supervisor on time.
- Prepare the meeting(s) with your supervisor.
- Suppress the idea that you can prepare a talk in a day or two.

Some hints

Hints

- Take your time to understand your literature.
- Be proactive! Look for additional literature and information.
- Discuss the content of your report with other students.
- Be proactive! Contact your supervisor on time.
- Prepare the meeting(s) with your supervisor.
- Suppress the idea that you can prepare a talk in a day or two.

We wish you success and look forward to an enjoyable and high-quality seminar.